Application of agile methods in distributed software development

Ally Kuziwa
Kazibwe Flavia Ndagire

Department of informatics
Master of Information Technology Management
Master thesis 2-year level, 30 credits
SPM2016.21
Abstract

Agile software development methods are characterized by developer’s empowerment through self-organization, incremental requirements and constant knowledge sharing. Despite some effort to elaborate how successfully agile methods can be applied in co-located development teams, little is known on how the same is done when the development teams are distributed. Our research focus is to uncover factors that influence the successful application of agile methods in distributed development teams using a software development company as our case study. With three distributed teams applying different agile methods, we conducted focused ethnography alongside interviews to comprehend what and how various factors under distributed development teams influence successful agile methods application. It seems, all that matters is team structure, team spirit, effective communication, product requirements, and knowledge sharing. The team structure has more influence than other factors and therefore our suggestions for successful application of agile methods in distributed software development is to focus on the development team, above all factors. The analysis shows that the development methods themselves have little to do with the successful application of agile methods in distributed software development teams.

Keywords: Agile methods; distributed software development (DSD) teams; agile manifesto; team structure;

1. Introduction

It has been identified as a 21st century trend that many software companies are employing distributed software development teams using agile methods (Fowler, 2006). This trend promises benefits such as development cost reduction, access to skilled labor, shared knowledge and shorter time to market. There are a number of reasons as to why companies employ agile methods, but most important is the need to tackle the changes in requirements at any stage of the development process and achieve fast product delivery (Moeeni, Sanchez & Vakha Ria, 1997) and quick solutions to software bugs (Sureshchandra & Shrinivasavadhani, 2008). It is because of the above mentioned reasons that agile methods such as Scrum, Kanban, eXtreme programming among others are considered to be favorable development methods for any software company seeking a flexible process of adapting to changing requirements (Boehm, 2002).

Unfortunately, in distributed software development (DSD), the team members are not only separated by physical locations but also by time and cultural differences. The barriers caused by the above separations are not agreeing with the agile methodology principle that emphasizes in
practice, that developers work best in a co-located place and an environment with as small number of team members as possible (Ramesh, Cao, Mohan & Xu, 2006). The co-location and small number of team members is vital for agile methods because agile methods recommend frequent face-to-face communication which cannot be achieved easily in distributed software development. Additionally, as the teams are geographically distributed, they work greatly in isolation challenged by difference in time zones and cultural differences (Smite, Moe, & Ågerfalk, 2010), loss of trust and collaboration issues. These, among other challenges inevitably impact the application of agile methods by developers and lead to development teams to become increasingly noncompliant to the agile methodology principles (Akbar, Haris, & Naeem, 2008).

Furthermore, agile software development involves small continuous iterations with constant product revision and validation as specified in the agile manifesto (Highsmith, 2001). This makes agile methods application by distributed software development teams complicated, despite the effort to have regular daily standup meetings in distributed teams to check for product revisions. The complication is due to the fact that there is still never a perfect time for all DSD team members to meet and iteratively discuss the progress of developed product, especially with big time differences. Also studies such as that of Leffingwell (2007) indicate that agile methods are unsuitable to be applied within large, globally distributed teams working on long lived projects, something that the company used as our case study is exactly doing. It is a large company, with distributed teams working on long lived projects and at the same time applying agile methods. This makes us wonder why and especially how companies such as our case company manage to successfully apply agile methods in DSD teams against all odds. We therefore intend to understand the secrets behind these unexpected successes. We want to know “what are the factors influencing successful application of agile methods in DSD”? We investigated work practices of DSD team members in a software company that develops long life cycle products and services to establish the factors behind successful application of agile methods in DSD.

It is evident that there are limitations of applying agile methods within distributed development teams, and the fact that agile methods remain widely applied in DSD, led us to the factors that influence successful application of agile methods in distributed teams. It is the complicated and questionable combination of the agile methods and distributed software development that motivated our investigation of the factors influencing success. Understanding these factors might on the other hand support or disagree with the success stories which explain that inspite of all the known agile methods limitations, distributed teams are doing just fine with the method (Sureshchandra & Shrinivasavadhani, 2008). Fact is, little is known about how software companies overcome these limitations (Turk, France & Rumpe, 2014), and therefore
establishing factors that influence the successful application of agile methods in a distributed software development teams is of importance to the field of distributed software development.

2. Related research

Studies such as that of McMahon (2005) go against the general argument such as that of Ramesh et al. (2006) which insists that agile methods only work best in small co-located development teams. In his work, McMahon (2005) explains the weakness in the argument by introducing customer value as described by the agile manifesto. He acknowledges the need for small development team and co-location but questions its validity in today’s world where one cannot guarantee full-time customer presence onsite and self-directed team skills. Onsite customer presence is emphasized in agile methods’ application to minimize misinterpretation of customer requirements (Highsmith & Cockburn, 2001). By applying agile methods in DSD, onsite customer presence becomes literally impossible because the development team is never on one site. As a result, DSD automatically triggers communication issues not only among developers but also between customer and the dispersed development team. In this study, McMahon (2005) similar to Nerur, Mahapatra & Mangalaraj, (2005) and Pikkarainen, Haikara, Salo, Abrahamsson & Still, (2008), recognizes communication as among the most important aspects of agile methodology principles. He thus proposes developer’s concentration on communication with the customer instead of among themselves in an effort to adopt with the DSD communication challenges.

The same can be seen from the study conducted by Paasivaara & Lassenius (2006) when they challenged if the global software development could benefit from agile methods application. Once again, Paasivaara & Lassenius (2006) as McMahon (2005) recognize the challenges of applying agile methods in DSD. Although Paasivaara & Lassenius (2006) send the blame directly to DSD and not agile methodology principles, the main discussion in both studies seems to advocate the idea that even without face to face communications, agile methods can still be adopted to fit into distributed development teams. It is clear that the studies try to defend the application of agile methods in DSD by explaining how effectively agile methods can be applied within distributed teams as also discussed by Ambler, (2012); Cohen, Lindvall, & Costa, (2004) and Phalnikar, Deshpande & Joshi (2009). The studies strengthen their argument by using successful case studies, but yet, the studies do not show or even predict the factors that led to the claimed successful application of these agile methods in distributed development teams.

Additionally, a lot of other studies including that of Abrahamsson, Salo, Ronkainen & Warsta, (2002); Holmström, Fitzgerald, Ågerfalk & Conchúir, (2006); Ramesh et al. (2006); and Sureshchandra & Shrinivasavadhani (2008) focus on how agile methods can be customized to fit
into distributed software development. Sureshchandra & Shrinivasavadhani (2008), for example, talk about agile methods customization as the only solution towards going parallel with indispensable DSD. The good news is that all the studies above do not see agile methods as a source of problem but rather a solution to the unavoidable DSD trend. It is due to the flexible nature of agile methods that a successful application of the methods can be achieved in DSD. We recognize that the studies argue that being agile is the ability to be flexible enough to accommodate all necessary changes in development events, and that agile methods customization is part of agility. Agility as discussed by Fowler & Highsmith (2001) is the power of agile methods to handle unpredicted changes. Since the agile manifesto is silent on how much customization it can accommodate, the studies recommend that any change can be made to the agile methods as long as they do not contradict the agile manifesto. It should be noticed that the Agile Manifesto as explained by Beck, Beedle, Van Bennekum, Cockburn, Cunningham, Fowler & Kern (2011) and Cockburn (2006) consists of four principles (value statements) which state that:

● We value individuals and interactions over processes and tools.
● We value working software over documentation.
● We value customer collaboration over contract negotiation.
● We value responding to change over following a plan.

Therefore, looking at the fourth principle, the agile manifesto seems to encourage “responding to changes”. To summarize the principles, we believe it is acceptable for agile methods to be customized so long as there is a developer-customer interaction to deliver the desired product. In short, what the agile manifesto emphasizes is fulfilling customer requirements. Therefore, the discussions from the above studies are great starting points in explaining how agile methods can be applied in distributed teams to deliver the needed product. However, there is still a need to establish a basic foundation on how agile methods’ application can be done successfully in DSD. In other words, although the studies explain how to apply agile methods in DSD, software companies might fail to do a successful application such as that of McMahon (2005) and Paasivaara & Lassenius (2006) if they do not know the factors behind these successes. This is why it is important to identify the factors behind the successful application of agile methods in distributed software development to guide agile practitioners into deliberate successful application of agile methods in DSD.

Additionally, studies such as that of Wan & Wang, (2010) recognize the need to understand the factors that could lead to application of agile methods in a successfully better way, or as what Chow & Cao, (2008) calls critical success factors in software projects. The weakness in these factors is that they are mostly associated with top management and that they are method-specific as explained by Power, Sohal & Rahman (2001); Misra, Kumar & Kumar, (2009); and
West, Grant, Gerush, & D’silva, (2010). Besides, most of the arguments above are in co-location environments and supply chain management contexts and not in a DSD context. Thus, our identified factors can stand as guidelines to both researchers and practitioners on explaining how and possibly why agile methods are still widely applied in distributed development teams, inspite of all the known limitations. They can also help the general agile community to see how agile methods through work practices have been adapted to fit these distributed development teams. We will look closely at the work practices in terms of their usefulness, priority and how they are deployed (styles) by different distributed development teams. The results can serve as guidance for organizations considering practicing agile methods in distributed software development teams.

3. Methodology

3.1 Empirical research methods

As a discipline, software development is not only about developing technical solutions, but also incorporates aspects about work practices such as team organization, management and performance. Our research question will hopefully be answered through studying the work practices of distributed teams in a technical setting. Some of the empirical research methods that we considered included controlled experiments, case studies and surveys. Among the three mentioned methods, we used a case study as it was most suitable for gathering knowledge about activities, events that happen to individuals or an entire team being investigated (Yin, 1989). In order to learn more about how agile methods are applied in distributed teams, our approach was to use an exploratory case study of a single company to observe and monitor activities and events of three distributed teams within a software development industry setting through formalized investigations.

3.2 Research site

In this section, we describe the company that we studied from December, 2015 to March, 2016. During the three months, we had about 10 site visits, a number of face to face meetings, made Skype calls, attended team activities and development events and exchanged emails with the teams. The company which we have renamed Olympus for anonymity has been one of the largest suppliers of IT services in Scandinavia for over 30 years. Olympus delivers products and services to the global market in more than 20 countries with headquarters in Finland and regional offices in Sweden and India. It employs around 14,000 persons. The employees are distributed in 20 different countries supporting various product development and maintenance of full lifecycle IT services. These services include business process services, application services,
development of application platforms and outsourcing of IT services. Since it was founded in 1968, Olympus employed the waterfall methodology which favors sequential software development with predefined requirements. However this methodology had drawbacks in meeting changing customer requirements over the lifecycle of IT services. As a result, the company sector that we studied adopted and has been using agile methods since 2007 to deal with requirement uncertainty and to realize faster development times.

3.3 Sample selection

In this section, we explain the selection of development teams investigated in the study. The study took place in Sweden, where one of the Olympus’ sectors, the HCWE (Healthcare, Welfare and Education) sector is located. HCWE is focusing on developing products and services for the Nordic Social Service domain. It has a strong market position and is a market leader in Finland, Sweden and Norway in the selected domains; Education, Elderly Care & Family care. The HCWE sector has development teams which use agile methods to develop and maintain long life-cycle products, which makes the sector a good study case in reference to Leffingwell’s (2007) observations that agile methods are unsuitable to be applied within distributed development teams working on long lived projects.

Olympus has many development teams that are using different agile methods that were suitable for the study as vast information sources. Once we had decided on which sector of Olympus to investigate, we applied the purposive sampling technique to select the three development teams to represent the other teams within the sector. The three teams were chosen because they had distributed team members and were working on different long life cycle products using different agile methods.

<table>
<thead>
<tr>
<th>Team</th>
<th>Sweden</th>
<th>India</th>
<th>Team size</th>
<th>Agile method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team 1</td>
<td>Developers</td>
<td>Developers</td>
<td>10 members</td>
<td>Scrum</td>
</tr>
<tr>
<td></td>
<td>Scrum Master</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Area Product Owner</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team 2</td>
<td>Developers</td>
<td>Developers</td>
<td>14 members</td>
<td>Kanban</td>
</tr>
<tr>
<td></td>
<td>Scrum Master</td>
<td>Scrum Master</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Area Product Owner</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team 3</td>
<td>Developers</td>
<td>Scrum Master</td>
<td>14 members</td>
<td>Hybrid method</td>
</tr>
<tr>
<td></td>
<td>Scrum Master</td>
<td>Developers in a sister-team</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Area Product Owner</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
It was an advantage for the study to have three teams practicing three different agile methods; Scrum, Kanban and Hybrid because the teams would provide us with vast data on agile methods application. The hybrid method used by Team 3 is a combination of Scrum and Kanban. Team 3 works with developers from another team, referred to as a sister-team which is an independent team from India. The developers in the sister team are not directly integrated with the team members of Team 3 in Sweden. This means, that there are two different teams, one from India and another from Sweden connected and communicating with the same area product owners. Area product owners are people with enough domain knowledge to specify what work needs to be done by both teams and leave it to the teams to decide on how the development work is done.

The participants from the teams were selected based on criteria such as their experience working with distributed agile teams and their roles in the distributed team. The participants had different roles such as team managers, developers, scrum masters and area product owners working in two countries, Sweden and India, within distributed development teams. The different roles were included to get an overall perspective from the three distributed development teams using different agile methods (see table 1). For the study purposes, the participants were selected from different teams, from two product development sites in Sweden and India. Based on the participants’ work experience and in-depth knowledge about agile methods, we are confident that the selected sample accurately represents the application of agile methods within distributed teams in Olympus.

### 3.4 Data collection

Our data collection process involved various activities organized into three different phases. The phases were Informants Interviews, In-depth-Interviews and Focused Ethnography.

#### 3.4.1 Informants Interviews

The informants interviews were probing interviews with semi-structured questions to gather clear understanding of the company structure, work processes and development procedures. It was important to find out if Olympus as a software company would fit as a possible research site. We interviewed one Team manager and three developers from Sweden in person, and two developers from India using Microsoft Lync, for about an hour each as seen in table 2 below. The interviews helped us to get a clear picture of what the company profile is in terms of its development methods, team structure, work practices and the ways in which it met our research objectives. The five developers interviewed belong to different software development teams and were knowledgeable about the company profile since different teams are working for different software products in different styles (way of working) as later discovered. To ensure that we got
balanced views, two of the five developers that we interviewed worked in India. We therefore used the informants interviews to narrow down our research focus, shaping our study design and mostly important polish our research gap.

<table>
<thead>
<tr>
<th>Participant’s Code</th>
<th>Location</th>
<th>Role in the team</th>
<th>Interview Duration</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pilot 1</td>
<td>Sweden</td>
<td>Team Manager</td>
<td>51:39</td>
<td>19/01/2016</td>
</tr>
<tr>
<td>Pilot 2</td>
<td>Sweden</td>
<td>Developer</td>
<td>57:55</td>
<td>22/02/2016</td>
</tr>
<tr>
<td>Pilot 3</td>
<td>Sweden</td>
<td>Developer</td>
<td>59:27</td>
<td>25/02/2016</td>
</tr>
<tr>
<td>Pilot 4</td>
<td>India</td>
<td>Developer</td>
<td>43:13</td>
<td>25/02/2016</td>
</tr>
<tr>
<td>Pilot 5</td>
<td>Sweden</td>
<td>Developer</td>
<td>54:54</td>
<td>25/02/2016</td>
</tr>
<tr>
<td>Pilot 6</td>
<td>India</td>
<td>Developer</td>
<td>52:14</td>
<td>18/02/2016</td>
</tr>
</tbody>
</table>

Table 2. Informants interviewee’s table.

3.4.2 In-depth Interviews

In this phase, we conducted face-to-face thorough interviews with three different development teams. A total of 21 participants were interviewed and recorded for later references. A semi-structured questionnaire was used for in-depth interviews. The informants’ interviews informed the design of the questions in this semi-structured questionnaire (see appendix 1) as explained in section 3.4.1 above. We divided the questionnaire into sections such as introduction, team structure, development method, work process and communication feedback, and version control for clarity and better analysis. For better understanding of the work processes within the development methods, having open-ended questions as suggested by Louise & While (1994) provided us with extra information from respondents which proved to be useful in our analysis as shall be seen later. Participants in In-depth-Interviews did not involve management due to the fact that we were only concerned with development methods and only development team members could give us the information we needed. Various roles in the development team from both India and Sweden were interviewed as seen from the table 3 below.

<table>
<thead>
<tr>
<th>Participant’s Code</th>
<th>Location</th>
<th>Role in the team</th>
<th>Interview Duration</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interviewee</td>
<td>Country</td>
<td>Role</td>
<td>Duration</td>
<td>Date</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
<td>-------------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>Dev-01-Team 1</td>
<td>India</td>
<td>Developer</td>
<td>52:14</td>
<td>18/02/2016</td>
</tr>
<tr>
<td>Dev-02-Team 1*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dev-03-Team 1</td>
<td>Sweden</td>
<td>Developer</td>
<td>51:39</td>
<td>19/02/2016</td>
</tr>
<tr>
<td>Apo-01-Team 1</td>
<td>Sweden</td>
<td>Area Product Owner</td>
<td>57:55</td>
<td>22/02/2016</td>
</tr>
<tr>
<td>Sm-01-Team 2</td>
<td>Sweden</td>
<td>Scrum Master</td>
<td>59:27</td>
<td>25/02/2016</td>
</tr>
<tr>
<td>Dev-01-Team 2</td>
<td>India</td>
<td>Developer</td>
<td>43:13</td>
<td>25/02/2016</td>
</tr>
<tr>
<td>Dev-02-Team 2</td>
<td>Sweden</td>
<td>Developer</td>
<td>54:54</td>
<td>25/02/2016</td>
</tr>
<tr>
<td>Dev-04-Team 1</td>
<td>Sweden</td>
<td>Developer</td>
<td>51:11</td>
<td>25/02/2016</td>
</tr>
<tr>
<td>Dev-05-Team 1</td>
<td>Sweden</td>
<td>Developer</td>
<td>49:25</td>
<td>26/02/2016</td>
</tr>
<tr>
<td>Sm-01-Team 1</td>
<td>Sweden</td>
<td>Scrum Master</td>
<td>51:56</td>
<td>26/02/2016</td>
</tr>
<tr>
<td>Dev-03-Team 2</td>
<td>Sweden</td>
<td>Developer</td>
<td>55:42</td>
<td>26/02/2016</td>
</tr>
<tr>
<td>Dev-04-Team 2</td>
<td>India</td>
<td>Developer</td>
<td>47:53</td>
<td>26/02/2016</td>
</tr>
<tr>
<td>Dev-06-Team 1</td>
<td>Sweden</td>
<td>Developer</td>
<td>38:41</td>
<td>26/02/2016</td>
</tr>
<tr>
<td>Dev-05-Team 2</td>
<td>Sweden</td>
<td>Developer</td>
<td>56:57</td>
<td>01/03/2016</td>
</tr>
<tr>
<td>Sm-01-Team 3</td>
<td>Sweden</td>
<td>Scrum Master</td>
<td>49:42</td>
<td>01/03/2016</td>
</tr>
<tr>
<td>Dev-07-Team 1</td>
<td>Sweden</td>
<td>Developer</td>
<td>53:44</td>
<td>01/03/2016</td>
</tr>
<tr>
<td>Apo-01-Team 2</td>
<td>Sweden</td>
<td>Area Product Owner</td>
<td>40:04</td>
<td>01/03/2016</td>
</tr>
<tr>
<td>Dev-08-Team 1</td>
<td>Sweden</td>
<td>Developer</td>
<td>44:19</td>
<td>03/03/2016</td>
</tr>
<tr>
<td>Dev-01-Team 3</td>
<td>Sweden</td>
<td>Developer</td>
<td>52:54</td>
<td>03/03/2016</td>
</tr>
<tr>
<td>Dev-02-Team 3</td>
<td>Sweden</td>
<td>Developer</td>
<td>36:26</td>
<td>04/03/2016</td>
</tr>
<tr>
<td>Dev-03-Team 3</td>
<td>Sweden</td>
<td>Developer</td>
<td>47:22</td>
<td>04/03/2016</td>
</tr>
</tbody>
</table>

* Two developers from India were interviewed at the same time.

Table 3. In-depth interviewee’s table.
3.4.3 Focused Ethnography

Focused Ethnography is a qualitative short time observational research method. We used this observation method to physically observe development events in parallel with follow up questions in order to achieve data intensity in a short period of time as recommended by Knoblauch (2005). It was important for us to observe these development events through the daily routines of the team members to see how they apply agile methods in a natural setting. These events include daily morning standups, triages, refinements, retrospectives, cross-teams, sprint planning and product demo meetings. Luckily, we had full access to the activities of the three development teams. Although we tried to be with the observed team during their working hours from around 8:00 am to 5:00 pm, we decided to concentrate in specific development events that involved all team members working both in Sweden and India. We had at least one week of observation time for each team and various development events were observed and recorded during that time using an observation questionnaire (see appendix 2). Despite the fact that this observation was done in Sweden only, we are confident that the quality of the data collected was good and equally represented both Swedish and Indian team members. This is because of two major reasons; first, the observed events were attended by both the Swedish and Indian teams through audio-visual communication tools. Second, two team members working in India were physically present in Sweden for site visit during the observation period.

However, the observed development events took place in Sweden at different times as clarified in table 4a, 4b and 4c below. It is important to note that we had to separate the below tables because some teams had special development events according to the development method they use, team style and the agreement of their “way of working”.

<table>
<thead>
<tr>
<th>Observed Event</th>
<th>Frequency</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morning Standup</td>
<td>Daily</td>
<td>09:00 am</td>
</tr>
<tr>
<td>Sprint Planning</td>
<td>After every other week</td>
<td>Varies</td>
</tr>
<tr>
<td>Refinement</td>
<td>After every other week</td>
<td>Fridays</td>
</tr>
<tr>
<td>Retrospective</td>
<td>After every two weeks</td>
<td>Varies</td>
</tr>
<tr>
<td>Demos</td>
<td>End of the month</td>
<td>Fridays</td>
</tr>
</tbody>
</table>

*Table 4a. Team 1 development events.*
Team 2.

<table>
<thead>
<tr>
<th>Observed Event</th>
<th>Frequency</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morning Standup</td>
<td>Daily</td>
<td>08:45 am</td>
</tr>
<tr>
<td>Retrospective</td>
<td>After every two week</td>
<td>Varies</td>
</tr>
<tr>
<td>Refinement</td>
<td>After every other week</td>
<td>Varies</td>
</tr>
<tr>
<td>Triage</td>
<td>Almost Daily</td>
<td>08:30 am</td>
</tr>
</tbody>
</table>

*Table 4b. Team 2 development events.*

Team 3.

<table>
<thead>
<tr>
<th>Observed Event</th>
<th>Frequency</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morning Standup</td>
<td>Daily</td>
<td>09:00 am</td>
</tr>
<tr>
<td>Refinement</td>
<td>After every other week</td>
<td>Varies</td>
</tr>
<tr>
<td>Cross-team</td>
<td>Every Tuesdays and Thursday</td>
<td>10:15</td>
</tr>
<tr>
<td>Retrospective</td>
<td>After every two week</td>
<td>Varies</td>
</tr>
</tbody>
</table>

*Table 4c. Team 3 development events.*

The following is a brief explanation about the development events from table 4a, 4b and 4c. **Morning standup:** This is a daily morning meeting where every team member mention and if necessary describe what they have worked with the day before. The meeting serves as a feedback mechanism to offer or ask for help if a member deem it necessary. **Refinement:** This is an after every other week meeting organized by the Area Product Owner (APO) to introduce or take out product items as per the need. It is a refinement to the product developed and it is used even for maintenance purposes to prioritize the bugs that need fixing. **Retrospective:** It is a meeting that takes place twice a month with the aim of evaluating how far the team has been able to accomplish the targeted objectives. **Demo:** This is a demonstration meeting with the aim of trying out new features and showing them to all team members. Demo is conducted for knowledge sharing purposes since different product items are developed by different developers in the team. **Sprint Planning:** It is a scrum method event which is used to divide the items into different sprints. Sprints are small development phases of the scrum development method where objectives are redefined from previous sprints. **Triage:** This is a special morning meeting
used by Team 2 to discuss customer feedback through the support team. The support team in Team 2 serves as the first line of communication with customers since Team 2 is doing maintenance of a product already in the market. The frequency of the meeting is almost daily because there is always a complaint of some kind from customers about the product. When there is nothing to discuss, the meeting does not take place, as it happened once during our observation time. Cross-team; This is a Team 3 meeting where representatives from teams working on the same product line (upper-secondary education) meet to discuss their experience with the product. It involves the sister team in India and it aims on simplifying the work by telling each other what one team has been facing so that other teams can be aware when facing the same issue.

It should be noted that, although some of the events above seem to be happening after every other week or two and therefore seem to require more observation time than a week, it was made possible by the fact that two teams were sitting on the same floor and that we could freely move between teams. Also, movement amongst cross-teams was possible because the teams conducted the development events on different days and even when the days collided, the events were not happening at the same time.

### 3.5 Data analysis method

In this section, we explain the techniques, processes and procedures that we used to define and interpret the collected data from the interviews and observations. Our research process resulted in collection of qualitative data from 21 participants that we interviewed and recorded. The in-depth interviews were manually transcribed and the informants’ interviews were used for background knowledge gathering. Qualitative content analysis was used to analyze the data by identifying patterns and concepts extracted from interviews and observational data by conventional content analysis approach. Conventional content analysis approach is one among three qualitative content analysis approaches as described by Hsieh & Shannon (2005). It is an approach that is used to analyze data to produce codes which are grouped into categories before conceptualizing the categories into meaningful concepts (Kondracki et al., 2002).

Each in-depth interview of a participant was analyzed according to the team that they belong to since each team was using different development methods and different style of working. The iterative process of conventional content analysis made it possible to identify key concepts that emerged from each individual interview. We coded the transcripts into overarching categories then explored associations amongst the categories such as comparing the team’s spirit (working morale) against the development method used, the product developed and the practices of the developer working within a distributed team. The process, therefore, helped us understand how agile methods are applied within the company sector from a participants’ perspective.
The conventional content analysis approach was in short used to interpret the collected data in order to identify important patterns. To illustrate this, in the case of the in-depth interviews, examples of categories identified are product requirements, team spirit, choice of method, team freedom, team self-organization, team experience, flat team structure. We went on and dissolved some categories within other categories and extracted others from already existing categories to form concepts.

Figure 1 shows an overview of the coding process from categories into a concept

The team structure as a concept was derived from combining team freedom, self-organization, team experience, and flat team structure categories (See Figure 1). The same approach was used for focused ethnography data whereby field notes were coded, categorized and conceptualized. It is these categories that are used to explain the factors that influence successful application of agile methods in DSD teams.

3.6 Limitations and ethical considerations

Our research method is somehow limited by the manner in which observations of development teams were conducted during focused ethnography. Due to obvious reasons we were not able to observe both the Swedish and Indian sides of development teams at the same time. We therefore decided to physically observe only the Swedish part of the team. However, we used collective development events which involved both parts of the team to observe the Indian side. Digital communication tools such as Microsoft Lync were used to facilitate the Indian part observation during those collective development events.
Also, some of the observed development events in Team 3 that did not involve the developers in India were conducted in Swedish. This did not affect our research process since we have basic knowledge of the Swedish language, although we are both English speakers. We still attended the events and asked for clarifications where and when necessary. The good thing is, for the teams 1 and 2, the working language was English.

Ethically, we used a coding mechanism to ensure anonymity of the participants in the entire research process. The purpose of the study was clearly explained to the respondents in order receive their consent to participate. The anonymity of the participants was achieved by giving each of them different codes so that any of the identifying information such as contact details and names remains confidential.

The results of the study are relative to the context and environment within which the investigated development teams operate; as such statistical generalizations cannot be made from our single case (Flyvbjerg, 2006). Our results were conceptualized inductively. In other words, the data collected from our participants was analyzed basing on the observations and interviews about the work practices from our case; that is the Healthcare, Welfare and Education sector at Olympus. The results might not be generalized for all software companies like Olympus but can support analytical generalizations and generates new knowledge about the application of agile methods in distributed software development teams.

4. Findings and analysis

This section presents the five fundamental factors that are the findings from our study. We described the factors with simple analysis before discussing them in the next section (section 5). The factors regarded as fundamental to the successful application of agile methods in DSD are team structure, team spirit, effective communication, product requirements and knowledge sharing as explained below.

4.1 Team structure

One of the important factors that influence successful application of agile methods in DSD is the team structure which is defined through roles and responsibilities. To understand team structure through roles and responsibilities, the figure below represents the overall structure of all the three distributed development teams studied.
Figure 2. An overview of a distributed agile team at the HCWE sector at Olympus.

The main team member roles are the area product owner, the scrum master and the developer as described in table 5 below.

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area Product Owner</td>
<td>*Represents the customer * Knowledgeable about business laws and regulations * Expert on finding customer needs * Monitors changes throughout the lifecycle of the product * Prioritize tasks in the team * Stands as primary tester * Responsible for the product.</td>
</tr>
<tr>
<td>Scrum Master</td>
<td>*Organizes team meetings and facilitate development events such as demos, sprint planning, retrospective and refinement * Monitors team progress ensuring that tasks are performed on schedule * Guardian to team documents such as team agreements and product definitions.</td>
</tr>
<tr>
<td>Developer</td>
<td>*Codes and debugging software * Runs code through checks and verification to ensure quality.</td>
</tr>
</tbody>
</table>

Table 5. Roles and responsibilities of team members.

Team 1 is responsible for developing a Mobile application for a desktop product. The developers in Team 1 are located in Sweden and India. They are responsible for product features development, bug fixing and testing code. For the development process, the team uses Scrum.
and begins their work each day with daily standups. To share knowledge, the team uses Microsoft Lync software for video and audio calls to communicate to each other. Information about tasks is documented in the digital scrum board created in the Team Foundation Server (TFS) which all team members both in Sweden and India have access to.

**Team 2** is responsible for carrying out maintenance work on school administration systems for a municipality in Sweden. The team uses the Kanban agile method because their work largely depends on customer requirements that cannot be predetermined. Each morning, the team has “Triage meetings” occasionally to discuss new issues brought by product support staff. Product support staffs are part of team members whose job is to monitor queries brought by customers. When issues arise from a customer, the customer calls the support staff who take note of the issue, solve it if it is solvable and report it in the triage meeting the next day for revision by developers.

“The triage is where all reported issues from customers handles that the support didn’t handle themselves. It is not work items that is discussed, it is issues. The issues into the meeting can be either a bug, a customer request for new or changed functionality or a request for educational help. If the Triage members decide that the issue is an Bug, then an Bug item in TFS will be created and prioritized against other bugs and the developers start to work on it (when they have time). The support will not report the incident in the Triage if they already have solved it (unless in very unusual findings). The Triage is for fast decisions for issues that cannot be handled in first line support” (Pilot 1, email).

Before the study, the triage and standup meetings were conducted separately whereby the team members in Sweden had their own meetings and the members in India had theirs. This complicated the understanding of the tasks by the team because the connection between team members of the two sides was through the area product owners only. Team 2 is the team with the largest number of Area Product Owners (APO) among all three teams. It has 5 APOs who are in charge of various school systems all working together on the customer requirements. The team conducts joint “triage meetings and standup meetings”, and shares information about their work in the Team Foundation Server (TFS) which is also accessible to the members in India.

**Team 3** is responsible for developing features in a Learning Management System (LMS) for compulsory level schools in Sweden. The product features of the LMS are developed by teams both in India and Sweden. For its development method, Team 3 practices a combination of Scrum and Kanban which we have referred to as Hybrid method. The team uses a physical board to share knowledge about tasks using story maps because there is no direct sharing of tasks between the Indian and Swedish teams. Team 3 only uses the digital board for tracking and indicating work items which are ready for testing.
Generally, we found out that all the three teams are self-organized and responsible for coordinating their work. They also maintain a flat team structure with unclear separation of roles amongst the team members. “We as a team, in our daily work have a lot of control over the product developed and assigning work items within the team”, a developer (Dev-03-Team 3) emphasized. So, from the interviews, it was clear that the teams have complete authority and responsibility over all aspects of their development work such as planning decisions and the product being developed. However, this doesn’t mean that the teams are uncontrolled by management. The Team Manager still establishes enough checkpoints to monitor team progress in all three teams. As confirmed by the Team manager in India through email, “I have scheduled monthly discussions with the teams where we do a retrospective of their monthly goals.” Additionally, summaries of the work breakdown structure from the Team backlogs are visible in the team rooms for the team manager to see. This was confirmed by the Team manager in Sweden (Pilot 1) when he said, “I have face-to-face meetings with all of my personnel every month and I get a good vision of how things are going in the teams.”

4.2 Team spirit

Another interesting finding is the observed high working team spirit where work related items were studied for each development team. By team spirit we mean what motivates the team members to work as a team, how they define their team roles, how they make decisions, who is considered a team member etc. Different development events gave us insights on the team spirit since most of the work items are discussed and created during the team’s development events. Documents such as “team agreement” and “way of working” which explain the do's and don'ts of a particular team showed similar patterns from one team to another. To our surprise, the mentioned documents are not based on the Scrum, Kanban or Hybrid methods used by the teams. They are rather derived from pure team spirit and social relations. As one scrum master (Sm-01-Team 1) said “We do not have a specific manual for the method we use, it is just an agreement on how we are working.” Other documents such as the definition of done (DOD) which describes when a work item should be considered done and complete, are also team spirit-dependent. For instance with its “way of working” agreement and the “definition of done” (DOD), we found out that Team 1 is self-organized, with high team working spirit, involving every available member’s decision on all its development events despite the fact that 80% (8/10) of team members are located in Sweden. The same was observed during Team 3’s retrospective meeting and Team 2’s refinement meeting whereby in order to meet deadlines, team members help each other to finish tasks before moving onto new tasks. “We usually work in pairs to complete a work item. In the case of absence of a member at work, someone else can join and we

† Some of the quotes have been slightly structured for clarity.
continue to work together. This ensures that information about the work item is not just for one person” as explained by one developer (Dev-02-Team 3).

4.3 Effective communication

Effective Communication is another factor that influences successful application of agile methods in DSD. As well known, communication is a big challenge in most software development industries as specifically mentioned by one developer (Dev-05-Team 2), “When combined with geographical, cultural and language barriers communication can itself determine a successful development of a given product.” We realized that working with distributed software development teams intensifies the communication challenge as the teams are limited by time zone differences. For example, the team members in Sweden and India have about only 5 working hours each working day to communicate with each other. “Since we are ahead in time, meaning that our day in India starts earlier than in Sweden, we have to wait for the area product owners to come in for work and then verify the work submitted in the morning before it is sent for testing,” said Dev-01-Team 2 when asked about delays in the verification process caused by time zone differences. We found out that Olympus is using Microsoft Lync communication software to address communication challenge. It is by addressing the communication challenge effectively that can influence successful application of agile methods in DSD.

4.4 Product requirements

The requirements of the product under development or maintenance also play a great role in influencing the successful application of agile methods in distributed development teams. While the decision to apply agile methods was made at management level and applies to the entire Olympus Company, it is the product requirements that determine what type of agile method is used by the team. For example, the development process of mobile application by Team 1 requires that the product is divided into features which need one or two weeks planning, development and refinements before the features are combined at specific allocated times. This pushes Team 1 to conduct sprint planning and demos which are Scrum methodology features. Sprints are important in Scrum development and although they seem like traditional sequential approach development, they recognize that user requirements cannot be fully understood and as developer 3 (Dev-03-Team 2) mentioned “emerging requirements can evolve at any time.”

Another example is the use of Kanban by Team 2. Team 2 is basically doing maintenance on school administration systems which were developed before. In maintenance, a number of software product issues commonly known as “bugs” come in either through customer feedback or complaints. Team 2 sorts the bugs and prioritizes them before fixing, but the frequency of the bugs is too unpredictable to plan ahead like in sprints. With distributed software development
team in mind, characterized by geographical and time zone difference it would be a nightmare trying to fix the bugs at random. This is why Team 2 picked Kanban methodology to ensure on time delivery according to the frequency and priority of the bugs. During the Triage meeting, the product support staffs inform the developers about the bugs reported by the customer. Some bugs are too critical to stay even a day longer without fixing from the time they are identified, for example login failure. “If the customer cannot login to the system, it is a problem that needs same-day fixing” Dev-03-Team 2 answered when asked “how long does it take to solve the bugs”? The point is, it is the requirements of the product under development or maintenance that dictates what agile method can be used by the teams.

4.5 Knowledge sharing
Since team members are geographically distributed, effective knowledge sharing mechanisms in Olympus play a part in ensuring the successful application of agile methods in DSD. At Olympus, knowledge sharing among right skilled personnel across time zones seems to be one of the main motivations for distributed software development as stated by developer 6 (Dev-06-Team 1) when he said “if it was not for knowledge sharing, we would not need the Indian developers.” In Olympus, the knowledge sharing happens in two levels; first within the local team and second amongst all team members. By local teams we mean teams located within the same geographical zone. During the study, two of the team members of Team 2 from India were on a site visit in Sweden for knowledge sharing purposes. The sharing of knowledge generally occurs through core members such as team mentors, distant members, consultants and agile coaches. These members use person-to-person daily interaction to learn or teach a required skill. They also have various established internal physical and digital communities, for example a community for all education-related matters, scrum masters community, agile community, user interface community among others. Additionally, they use pair programming mostly for local team members, and team blogs websites and Team Foundation Server (TFS) for all team members. “This ensures that information about the work item is not just for one person” as explained by one developer (Dev-02-Team 3).

5. Discussion
The main focus of our study was to investigate work practices of distributed software development teams during development events in order to identify factors influencing successful application of agile methods in DSD. By analyzing techniques, strategies and styles used to mitigate the challenges a software company faces when applying agile methods in distributed development teams, we identified the above interconnected fundamental factors
which influence successful application of agile methods in distributed development teams. The factors are interconnected in the sense that one factor relates and somehow depends on another factor as explained below.

5.1 Team structure
So, from the findings above, we can see how the flat development team structure at Olympus provides great freedom and control for the teams to find the best way to work on a given product. It is this freedom and self-organization that triggers team control and help the team to decide its “way of working” without considering any methodological principle. This is proved by developer 4 (Dev-04-Team 2) when she was asked “what can the team do without seeking permission from management”? She said “we can do almost anything we want on the product.” This is a double-edged sword that on one side compromises the agile methods principles but on the other side influences the successful application of agile methods in distributed development teams. When the development team decides what to do, when to do, and how to do with the product without method principle’s limitations or management interference, it makes it easy for the team to navigate through any development challenges caused by DSD. Therefore, freedom to defy method principles and considering flat structure and self-organization is a factor that influences the successful application of agile methods in DSD.

5.2 Team spirit
Also, from the findings, we noted that throughout the entire development phase, the high team spirit was very important in influencing successful application of agile methods in DSD. The fact that team members consider themselves as one and work on everything as a team, helped the teams to successfully complete development tasks even with absence of responsible team members. This enables the team to work as a family, correct each other’s mistakes instead of pointing fingers, and as a result facilitate on-time delivery of the developed product. Furthermore, the team spirit plays a role in reducing the cultural barrier by encouraging outside work team activities such as social events and therefore makes the two sides of the team feel the same. It is this feeling that the team is one family that influences successful application of agile methods in DSD. However, parallel with the agile manifesto (Cockburn, 2006), team spirit encourages work processes over documentation by allowing team members to randomly solve development problems without documentation. While this lack of documentation seems to be the right way, it risks the successful duplication of application of agile method in a different context. As confirmed by Pilot 1, “It is often I have talked to team members and asked them about the reason that things go so well at the moment. I often got the answer that they do not really know.” This means, when things go bad, the teams have no idea on how to go back to the
“good old days.” In other words, with no guidelines as to what happened when a method was used in a specific context, it will be so hard to replicate and successfully apply the same method in the same way by other practitioners in other contexts. Clearly, the documentation problem also is explained by the team manager (Pilot 1) to cause issues in the development process as follows;

“It is that if you only have the process in your head, then it is easy for others to have a slightly different view on the process. It is hard to commit to something that you have in your mind. And it is also hard to create improvements on a process that is unclear. And most of all... if you want to measure improvements on a process... how do you do that if you do not have it documented? How can you be sure the process is relevant the next time you measure it”?

5.3 Effective communication

Effective communication comes as part and parcel of agile methods application. This is because the agile manifesto emphasizes frequent interactions among developers and between developers and users. As Holmström et al., (2006) claim, the massive application of agile methods in distributed development teams proves to reduce distance issues through effective communication and coordination control. Although their argument is that effective communication is the result of agile methods application, the vice versa is more interesting. It is effective communication that influences the successful application of agile methods in DSD. This is due to the fact that agile methods emphasize communication above all other features during the development process. Since communication is the biggest challenge in DSD, Holmström et al., (2006)’s claim is that agile seems to be the best way to go to tackle communication challenges. This means, other traditional software development methods such as waterfall methods could prove to be more problematic in DSD. “Agile within distributed teams is a good way of working. I think it’s challenging to work with distributed teams. You could have waterfall and be in a bigger mess,” said Dev-03-Team 3. Waterfall for instance is characterized by phase-by-phase development style, which suggests that DSD team members would only meet to discuss the product at the end of the development phase. While this would be seen as an advantage for DSD members to do what they are supposed to do without interruption, it would enlarge the physical and working gap between DSD members especially at Olympus where all the area product owners for the three teams are located in Sweden. Thus, we believe that applying agile methods in DSD could be the best way to guarantee effective communication for successful application of development methods in DSD inspite of all known limitations.

Additionally, absence of physical face-to-face communication can be solved by introduction of digital tools. For instance, the use of Microsoft Lync for audio-visual communications at Olympus is what makes communication among the team members effective. Using Lync, the
development teams, Team 1 and Team 2 are able to conduct a face to face daily morning stand ups where all team members stand in front of a portable digital camera and mention their development status updates. Standup meetings do not require demonstration of any work item which means they could be conducted using audio calls only, but the all three teams are using video calls to ensure effective communication. “Facial expressions can tell a lot about someone in a meeting” as another developer (Dev-04-Team1) put it. Therefore, effective communication achieved by using audio-visual communication tools plays a part in influencing successful application of agile methods in DSD.

5.4 Product requirements

The requirements of the developed product seem to be dictating which and how an agile method can be used by the distributed development teams. As seen from the previous discussions, Olympus has no documents which explain what the teams should do or which development method to use. The choice of method and how to apply it relies on the product to be developed. This power of the developed product to dictate terms of development as supported by the agile manifesto (Cockburn, 2006) is one of the keys to successful application of agile method in DSD. Consider an example of a distributed development team with pre-defined way of working and a specific method to use for a given product. Obviously this distributed team will be restricted by methodological principles which might be too hard to fit with the given product. The opposite would give enough room for the team to mold the rules and principles in order to fit with the developed product. This can be proved by looking at Olympus as a software company that struggles to follow what the manifesto dictates, but at the same time its development teams seem to care more of what needs to be done by responding to changes rather than following the plan as discussed by Fowler & Highsmith (2001). SM-01-Team 2 noted, “We are not following the Scrum methods and principles all out. We are twisting it in the ways in which we feel works for us. Now with the maintenance work, it’s hard to work in sprints so we are leaning towards Kanban.” This is to say that Olympus like many other software companies tend to abuse and mold the agile principles to suit their projects and products environment instead of reluctantly be “agile”. While this might seem to be a compromise to the agile manifesto, it is actually a good decision which influences successful application of agile methods in DSD. It is a necessary evil caused by the nature of DSD itself which forces agile practitioners like Olympus to succumb to requirements of the developed product and have a sense of balance between agile principles and DSD challenges. As a matter of fact, the agile manifesto itself encourages any kind of compromise as long as customer needs are met (Beck et al., 2011). We cannot deny that the agile manifesto, as principles behind the agile methods, has been neglected by many software
companies (Jalali & Wohlin, 2010) that claim to be doing agile for the better application of these methods.

5.5 Knowledge sharing

For successful application of agile methods in DSD, knowledge sharing is also playing a very important role. As seen from the findings in section 4.5 above, knowledge sharing through various mechanisms such as communities makes distributed team members feel the need to do more by contributing and hence advancing their career through various knowledge sharing mechanisms. Team mentors, agile coaches and other consultants are among the key sources of knowledge within the development teams. One to one helping sessions are also taking place at Olympus where team members help each other to go through tough and challenging tasks. This was confirmed by Dev-03-Team 3, “During cross team standup, we meet virtually and use a webcam to talk across the teams. That way when they (referring to Indian developers) have run into a problem. I might be able to give tips to help them solve it or vice versa. We can share problems that we are facing.” This increases the working morale and reduces the chances of committing mistakes by working on challenging tasks. Pair programming is an additional technique mostly applied by the Indian team members to promote knowledge sharing especially when the task is new or tricky to one member. It automatically reduces the knowledge gap amongst team members and smoothens the development process. Thus, knowledge sharing motivates the team members to go an extra mile and undertake any development task necessary for the delivery of the product and automatically influence the successful application of agile methods in DSD.

6. Recommendations and Conclusions

It seems complete adoption to agile principles and use of the agile manifesto is affected by the work processes of the developers, and that many organizations claiming to being agile are influenced by their “way of working” decided by the team. Thus, we recommend that it is important to understand the working practices of a particular distributed development team in order to study adherence to agile principles in order to conclude that a particular distributed development team is applying agile methods successfully. This is to say that successful application of agile methods lies in the hand of a particular distributed development team and is defined by the working practices of that team.

Moreover, our study highlights that it is important to create a balance between the development method principles and development needs of a given development team. The development needs are continuously adjusted and require top management commitment to
allow the necessary flexibility. As can be expected with incremental requirements, the control over the development process should be left to the team. These efforts foster team spirit that build group practices such as knowledge sharing within communities of practice that lead to successful application of agile methods in DSD teams.

Apart from that, and due to the nature of the agile methods, we also recognize that current management practices need to be adjusted to meet the challenges of distributed software development teams for successful agile methods application. This means that not only agile methods’ practices must be adapted to fit into the needs of the particular organization (Amber & Lines, 2012) but also the organization through management has to be flexible enough to fit with the DSD trend. This can be done by for instance introducing frequent developer site visits across the teams in an effort for team members to familiarize with respective, working culture, traditions and ways of working. We also think that the management has to ensure that agile methods are customized only to address the challenges of DSD and increase customer satisfaction but not for the sake of customization.

To sum up, we suggest that agile practitioners fully avoid all unnecessary middle-man-dependent communications to foster full customer requirement understanding so as to reduce frequency of unexpected requirements. As seen from Olympus, the presence of area product owners as the only link between developers and customers create opportunity for misunderstood requirements. The situation is worse in DSD where remote team members depend gravely on area product owners who sit on one side of the team to define customer requirements. We understand the importance of some of these “middle men” as domain (product) experts especially if they are fully integrated in the team. But since the said expert is just one person (normally) with little if not non-technical knowhow, it is too risky to exclude developers during requirement definition phase with customers as explained by Pilot 1 when he emailed us about the roles of APOs.

“Some of the role description of an APO is not the way I see it. We have experienced that the hard way. Historically, the APOs have been the only persons in contact with our customers. And we have trusted them to make the right choices on what to build. That was quite successful as long as it was only administrators that were the end user of our system. But when new user categories started to use our applications, we realized the hard way that the APOs didn’t have any clue on how to build an intuitive application and how to solve customer needs. We had to let our developers meet the end users and let the team, instead of the APOs, decide on how to build the system (and how the system should work).”

Finally, we are convinced that the factors identified in this research have increased the knowledge about the application of agile methods and practices of distributed teams, particularly within teams using different agile methods in an organizational context. We believe
these factors can serve as guidelines to help companies mitigating challenges of DSD in software development in general and in using agile methods in particular. Although there seem to be no direct relationship between application of agile methods and DSD, the massive application of agile methods in distributed development teams indicates that times are changing and companies such as Olympus have pushed beyond the limitations and therefore it is vital to establish the secret behind these applications through the identified factors. The identified factors are the important ones and there appear to be other factors such as top management commitment, developer’s experience, among others which also influence the successful application of agile methods in software development teams as discussed by Chow & Cao, (2008); Power et al., (2001); Misra et al., (2009); Wan & Wang, (2010) and West et al., (2010).

References


Appendix 1. In-depth interview questionnaire

Interview code for respondent:  Start time:  End time:

Introductory questions
1. Tell us about yourself.
2. How long have you worked in software development? Probe for years of experience using different software development methods.
3. When did you join this company?
4. What products have you worked on in this company? Indicate product name and software development method that was used.

Study questions

A. Team structure
1. What is the name of the team that you currently work with?
2. Tell us about the product being worked on.
3. How long have you worked with this team?
4. Who assigns roles and responsibilities in the team?
5. Does your role determine the tasks that you are assigned?
6. Tell us about your role and responsibilities to the team.
7. Has your role and responsibility changed working on the product?
8. Do you have role and responsibilities in another product?
9. Do any of the team members have roles and responsibilities working on other products? Indicate the role and the product of that team member.

B. Development method
1. What development method is currently being used by the team?
2. How was the choice of the development method made?
3. Who participates in the selection of the development method?
4. Is it possible to use another development method?
5. What are the guidelines followed for the method and used by the team?
6. What guidelines of the method are not followed? Why?
7. What challenges are faced in applying the method?
8. How do you deal with the challenges in applying the method?
9. Who is responsible for making modifications to the method?
10. What modifications have been made to the method?
11. What are the reasons for the modifications?

C. Work As Tasks
1. Who is responsible for assigning tasks to the team members?
2. In what ways, do you think that the team is self-organized?
3. In what ways, do you think the team can make its own decisions on the product?
4. What tools are used to facilitate task assignment?
5. How are tasks monitored amongst the team members? What tools are used to monitor tasks?
6. How are the tasks implemented by the team members?

**Identify the role and responsibilities of each team member within the WBS**

1. What tools are used to monitor completed and pending tasks by the team members?
2. What unplanned activities are experienced by the team?
3. What is done to resolve or address unplanned activities?

**Communication, Feedback and Version Control systems**

1. What both formal and informal communication channels are used by the team members?
2. When and how often are these communication channels used?
3. What feedback systems are used amongst team members **during debugging**
   - to achieve product improvements across geographical locations
   - user acceptance testing
4. Who is responsible for the version control systems in terms of access control mechanisms?
5. What version control systems are used by the team to accomplish
   - record code revisions by developers
   - monitor versions across geographical locations
   - changes across geographical locations
Appendix 2. Observation questionnaire

Observation Follow-up Questions
1) Indicate the name of team activity to be observed.
2) Indicate activity duration.
3) What is the purpose for the activity?
4) How often is this activity carried out?
5) What stage of the product is being worked currently?
6) Observe the working environment within which the team works.
7) How many of team members are located in the office? Indicate the role of the team members.
8) Are their team members that are not located in the office? Indicate the role of the team members.
9) What tools are used to facilitate communication? Indicate formal and informal channels
10) What mechanisms are used to facilitate team interaction:
    1) between team members
    2) between team members working on different products
11) Observe the feedback systems used by the team:
    a) during debugging
    b) to achieve product improvements across geographical locations
    c) user acceptance testing
12) Observe the version control systems used by the team to:
    a) record code revisions by developers
    b) monitor versions across geographical locations
    c) changes across geographical locations

OBSERVATION GUIDELINES

Team structure:
  Identity-
  Location-
  Nature (co-located/distributed)-
  No. of members-
  Team spirit-

Work breakdown structure:
  Roles-
  Responsibilities-
  Control-
  Choices-
  Contingencies-
Communication & Coordination:
  When-
  How-
  Why-
  How often-